API Management Challenges in Ecosystems

Chapter · October 2019

DOI: 10.1007/978-3-030-33742-1_8

Sébastien Andreo
Siemens

Jan Bosch
Chalmers University of Technology

API management challenges in ecosystems

Sebastien Andreo[1] and Jan Bosch[2]

[1] Siemens AG Corporate Technology, Erlangen, Germany
sebastien.andreo@siemens.com
[2] Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden
jan.bosch@chalmers.se

Abstract. The API has become a cornerstone of software ecosystems, providing ways to drive innovation inside and outside the organization. Because of this criticality, we should manage APIs. The purpose of this study is to identify and classify the challenges that organizations evolving into internal ecosystems are facing as they have to deal with APIs. We performed a qualitative research study on three Siemens internal ecosystems with different sizes, technologies, and age. The results reveal that even if we are talking about the API economy, organizations are struggling with different aspects of API management related to Business, Architecture, Process, and Organization. The challenges identified in this paper provide a basis for future research.

Keywords: API, API Management, Case study.

1 Introduction

Not a day goes by that a company, or a governmental organization argues for or presents a plan to accelerate its digitalization and digital transformation. A cornerstone of those transformations is to make the digital services available to customers, and this is mostly realized using application programming interfaces (APIs). The development of APIs is not new and has been widely used since its inception in 1972 [1], but nowadays the monetization of API usage and the requirement to deliver software continuously to customers puts additional pressure on the development and the maintenance of APIs. As APIs are critical [2], any organization has to find measures to mitigate the risks of failure.

An API presents two sides. The first is a technical side and as a first definition we can use: an API is a technical answer to a business problem. The second is a business side because an API is a business enabler. In the simplest terms, we can define APIs as a set of requirements that govern how applications can interact and exchange data and how we want to deliver value to the customers. Because of the critical aspect of the APIs a form of management is required to mitigate the risks of failure. In this paper we consider the term API management as described by Jan Stafford in [3] where he compares, the API management to application life-cycle management: observing and controlling an API from creation to retirement.

In this paper, we highlight the challenges regarding API management in internal software ecosystems (ISECOs). To define an internal ecosystem, we used the definition proposed by [4]: an internal ecosystem is characterized by self-contained profit centers with their own business objectives, organizationally independent with their own product management, and have to a wide extent autonomous processes and software-engineering life cycles. To this end, we investigate the following research question to support our study: What are the major challenges for API management in internal ecosystems?

The challenges in this paper are based on the analysis of three internal ecosystems inside Siemens. The ecosystems present different aspects regarding the way they are integrating the different partners, the technologies involved, and also the size of the project itself.

The contribution of this paper is to provide evidence that even if the API economy has become a key driver of digital businesses, organizations evolving in internal ecosystems still face business and technical challenges to take advantages of this transformation.

The paper is organized as follows. The next section defines the research methodology we used to collect the challenges and presents the internal cases studied, while section 3 describes the challenges we identified about API management. We used the BAPO model [5] to cluster the challenges in order to reflect the impact of API management often considered as a technical challenge in all other product development concerns. Section 4 summarizes and orders the challenges. Then in section 5, we discuss threats to validity as well as implications for practitioners and researchers. In section 6, we provide an overview of related work. Finally, we conclude the paper in section 7.

2 Research Methodology

This section describes the research settings of the three ISECOs we investigated and outlines our research method.

2.1 Case Study Systems

The cases in this paper are based on the study of large ISECOs at Siemens. All case study systems, ISECO-A ISECO-B, and ISECO-C present a similar ecosystem structure, comprising a keystone which is a member of the software ecosystem that owns, operates and evolves a platform and multiple clients that build applications upon it. The ISECOs have established products in industrial and healthcare domains. However the ISECOs present distinctions in term of age, development size, technology and deployment, and finally in term of marketing: the keystone name is the market identifier, or the apps on top of the keystone are the identifiers. we summarized the characteristics in table 1.

Ecosystems	ISECO-A	ISECO-B	ISECO-C
# years of development	16	12	5
# deployment	client server	desktop	web
# of keystone devs	200	300	50-100
# of internal partners	6-8	6-8	5-10
# of external partners	2-5	5-10	5-10
# of apps	40-50	10-20	5-10

Table 1. ISECOs key data

2.2 Research Method

A participant observer approach and well-established case-study methods have been applied [6] [7]. The case data were collected directly from people working in different organizations. Principally two modes of collaboration were in place:

- Assessment: our role was to provide an external opinion and recommendations about the API management practices in place within the organization. To this end, we performed semi-structured interviews with relevant stakeholders: software architects responsible for the definition and evolution of the APIs and developers or key developers responsible for the implementation and the maintenance of the APIs. The interviews followed a guideline to guarantee that we collect the same areas of information from each interviewee, but also we leave space for a conversational approach to facilitate the interaction and to allow the interviewee to bring new topics to the discussion. If a new topic presented a high relevance for our assessment, we re-interviewed the previous stakeholders especially on that topic.
- Joint development: In this case, we worked closely with the organizations to develop tools and strategies to improve the API management in their existing software development landscape.

3 Challenges

In this section, we present our findings resulting from the study. We have chosen to cluster them according to the Business, Architecture, Process, and Organization (BAPO) perspectives [5] to reflect the impact of API management often considered as a technical challenge in all other product development concerns.

3.1 Business Challenges

As described in BAPO, the 'B' stands for Business, how do we generate revenue and profits.

Finding the optimal innovation speed for all partners In our internal ecosystem cases, the platform is the principal provider of new technologies, and

consequently, one of the innovation enablers. Its role is to provide an optimum speed to provide innovations to the partners without becoming a development bottleneck. To innovate means add, evolve, and change existing functionalities to create new customer added value. Unfortunately, each change has a cost: development cost for the provider and potentially a migration cost for the consumer. In the three cases we studied, we observed different strategies by the customers regarding the integration of the new platform's functionalities. From one side the innovators, who move as fast as possible behind the platform's intermediate releases. At the other end, the laggards who would postpone the migration as long as they can because they do not want it, do not have the resource to pay the migration or have other priorities. This heterogeneity comes from a divergence in the business model and the business goal as well as the staffing of the partners even if the domains are identical. From a business unit point of view, the platform can not favor one or the other partner type and has to compromise to satisfy both of them. The consequences are a slowing down the innovator partners and taking the risk they do not feel comfortable with "new" innovation speed and increasing the cost of development of the platform.

Finding B: Organizations suffer difficulties to provide an optimal innovation speed between all the ecosystem's partners by taking into account the heterogeneity of their business goals.

3.2 Architecture Challenges

As described in BAPO, the 'A' stands for Architecture and relates to technologies and structure to build the system.

Managing API dependencies Knowing which partner is using which API of the platform and how the partners are using these APIs is beneficial for the platform provider. It gives insight about the usage, the popularity and the criticality. Thereby, this data can help to get better decisions about the platform evolution direction.

In all cases, we encountered a similar pattern: a keystone provides a platform, and multiple clients build applications upon it. To separate the responsibilities and to encapsulate certain functionalities, architects have structured different layers of APIs and created public, internal, and in some cases, partner-specific APIs. If the platform is a kind of centralized application (like facebook.com) and the keystone provider controls it, it makes it easy to track the API dependencies between the platform and applications. However, if the platform is a shipping application, delivered as a set of binaries or if the development processes and release cycles are somewhat independent, the detection of undesired dependencies to the API surface becomes difficult or even impossible.

Finding A: In an internal ecosystem, maintaining an overview of the partners API usage gives the organization advantages to optimize its speed and customer added value.

3.3 Process Challenges

As described in BAPO, the 'P' stands for Process and relates to activities and way of working.

Handling of deprecation process As we indicate in the introduction, the API has a birth, a life, and a death: a life-cycle. If birth and life are unproblematic, the death implemented by a deprecation creates more struggles for organizations. We have noticed that finding the "good" date for a deprecation tends to be impossible and even more the date to altogether remove the API. Again the structure of the software ecosystem reveals several conflicting forces. From one side, we want to follow the ecosystem strategy to be able to create new business models and embrace the change and gain in speed. On the other side, we want to keep the current API stable, even if in our case not removing the API will cause additional costs for the maintenance of the software as well as potentially increase the complexity of the code and architecture. Which information and tools are missing to enable the customer to accept the deprecation? In [8] a beginning of an answer is provided: specifying the severity of change and the deadline or version of the deprecation. This, offers more transparency for the ISECO partners but, from the customer point of view, the effort of migration will still stay the same.

Finding P: A deprecation process is mandatory for the financial and technical health of the platform. When implementing it, it is difficult because of contradictory forces in internal ecosystems.

3.4 Organisation Challenges

Finally, the 'O' of BAPO, which stands for Organization, relates to teams and responsibilities.

Lack of education of the developers In the three projects at Siemens, we observed that these involve several technologies. This heterogeneity and multiplicity of technologies drastically increase the challenges for all the developers to understand those technologies and to master the subtleties. Another aspect of this lack of education is not related to the technology itself, but the awareness that manipulating APIs requires additional activities. At Siemens, we conducted worldwide dozens of internal training focused on API evolution, and the significant finding was that the ability to design and code an API is not depending on the seniority. Regarding the quality in term of usability and evolvability, the seniority has advantages, but we observed that a trivial aspect like controlling the

visibility of the API was often forgotten. Even if the developer is experienced, the trap is laid, API design and evolution need another mindset and another process.

Finding O: API design and evolution need a different mindset, awareness and continuous education of development teams to achieve better API quality

4 Relating findings to the cases

In this section, we will present the relevance of the finding for each case. We performed interviews with team architects and system architects for each project (3 to 4 per case) to characterize the importance of the identified challenges. After the presentation of a finding, the interviewees had to rank the relevance of the finding for their own case between 1 not relevant and 10, highly relevant. The average for each finding, and each case is presented in table 2.

	ISECO-A	ISECO-B	ISECO-C	Average	Std. deviation
B	7.5	8.7	7.0	7.7	0.9
A	7.0	8.7	8.3	8.0	0.9
P	9.0	8.7	6.0	7.9	1.7
O	9.5	7.3	8.3	8.4	1.1

Table 2. Findings relevance for each case

As depicted in table 2, for the three ISECOs, the relevance of the challenges are almost equivalent with a value around 8.0 of 10.0. The major difference observed is related to 3.3, when ISECO-A and ISECO-B indicate high relevance, ISECO-C tends to a medium relevance. ISECO-A and ISECO-B are comparable in term of development size and also in term of API Surface. Both have a wide API surface and different technology stacks to express APIs. Unlike ISECO-C, which has a narrow API surface and a single technology as API. The other explanation can be the age of the projects. ISECO-A and ISECO-B are older than ISECO-C.

5 Discussion

In the following section, we discuss construct, internal, and external validity threats.

Threats to construct validity is related to the relation between theory and observation. As we performed interviews, we relied on a potentially subjective statement of the participants. To reduce the effect, we listed only findings that have been mentioned several times. We also performed a two paths validation:

collecting the challenges in each organization, reformulating the challenge independently from the organization and requiring an evaluation of the relevance on the reformulated findings.

Threats to internal validity is related to a co-factors that may affect our results. In our case, interviewees might have given answers that do not fully reflect reality or have been exaggerated due to the current project stress situation regarding API Management. We worked for the three ecosystems for an extended period: two to three years, with different roles, assessor and co-worker, which gave us an overall measure and reduced the risk of an isolated measure. Furthermore, we anonymized and reformulated the challenges and proceeded to individual validation by the software architects and system architects.

Threats to external validity is related to the generalization of our results. In our case, the main thread is the representativeness of our cases. We investigated three Siemens ISECOs. The results can be specific to them. However, the collection of data was performed independently on several organizations.

6 Related Work

The problem of API management is not a new topic, but it has become more prominent in recent years. Study has already been performed [9] to evaluate the reaction of API evolution and to a deprecation. The need for continuous API Management is generally described in [10]. The authors propose several guidelines to balance the desire for agility and speed with the need for robust and scalable operations. Specifically to software ecosystems, Hammouda et al. [11] point out the necessity of a regular re-assessment of API architectural and design decisions to be able to balance the tradeoff between offering a current and modern API with offering a stable and backward compatible API.

Several tools and frameworks have also been developed to help the organizations to check/design their APIs. Lindman et al. [12] proposed a framework to help managers, designers, and developers to discuss API management. On a source code point of view, Brito et al. [13] proposed an APIdiff tool to detect syntactic breaking changes.

7 Conclusion

In this paper, we have given an empirical overview of the challenges, organizations implementing internal ecosystems are facing to realize effective API strategies. The empirical study was performed on three Siemens internal ecosystems. Even if in a first sense, an API can easily be viewed as something technical, we realized that many interconnections exist with other concerns of software development. We found 4 challenges, and we clustered around the BAPO model. The

main focus of the paper does not provide solutions, but rather highlights areas of improvements and open topics for further research.

The API economy is there, but to enable all organizations to benefit in a sustainable way from this possibility to innovate, further research and development is needed to increase the impact of APIs.

References

1. D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Commun. ACM*, vol. 15, no. 12, pp. 1053–1058, Dec. 1972. [Online]. Available: http://doi.acm.org/10.1145/361598.361623
2. J. Bloch, "How to design a good api and why it matters," *Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications - OOPSLA 06*, 2006.
3. "Why use new lifecycle tools in api management platforms?" [Online]. Available: https://searchmicroservices.techtarget.com/feature/Why-use-new-lifecycle-tools-in-API-management-platforms
4. K.-B. Schultis, C. Elsner, and D. Lohmann, "Architecture challenges for internal software ecosystems: A large-scale industry case study," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 542–552. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635876
5. F. J. v. d. Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering.* Berlin, Heidelberg: Springer-Verlag, 2007.
6. K. M. Eisenhardt, "Building theories from case study research," *The Academy of Management Review*, vol. 14, no. 4, p. 532, 1989.
7. G. Walsham, "Interpretive case studies in is research: nature and method," *European Journal of Information Systems*, vol. 4, no. 2, pp. 74–81, May 1995. [Online]. Available: https://doi.org/10.1057/ejis.1995.9
8. A. A. Sawant, M. Aniche, A. van Deursen, and A. Bacchelli, "Understanding developers' needs on deprecation as a language feature," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 561–571. [Online]. Available: http://doi.acm.org/10.1145/3180155.3180170
9. A. Hora, R. Robbes, M. T. Valente, N. Anquetil, A. Etien, and S. Ducasse, "How do developers react to api evolution? a large-scale empirical study," *Software Quality Journal*, vol. 26, no. 1, pp. 161–191, Mar. 2018. [Online]. Available: https://doi.org/10.1007/s11219-016-9344-4
10. M. Medjaoui, E. Wilde, R. Mitra, and M. Amundsen, *Continuous API management: making the right decisions in an evolving landscape.* OReilly, 2018.
11. I. Hammouda, E. Knauss, and L. Costantini, "Continuous api design for software ecosystems," in *2015 IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering*, May 2015, pp. 30–33.
12. J. Lindman, J. Horkoff, I. Hammouda, and E. Knauss, "Emerging perspectives of api strategy [unedited]," *IEEE Software*, pp. 1–1, 2018.
13. A. Brito, L. Xavier, A. Hora, and M. T. Valente, "Apidiff: Detecting api breaking changes," *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 507–511, 03 2018.